Projekt

Sterowniki robotów

Dokumentacja

Humanistycznie upośledzony robot akrobatyczny

HURA

Skład grupy:
Albert Lis, 235534
Michał Moruń, 235986

Termin: sr TP15

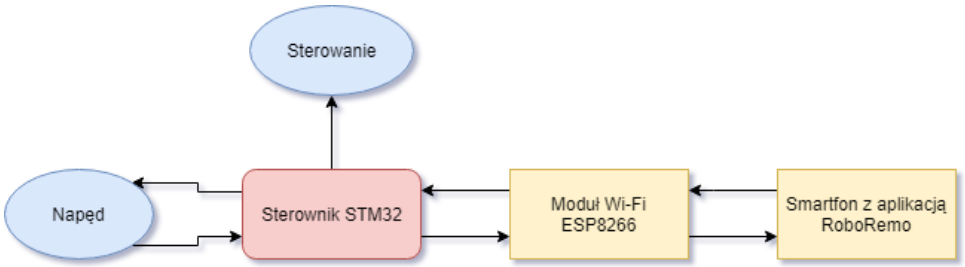
Prowadzący:
mgr inż. Wojciech Domski

4 czerwca 2019

Spis treści

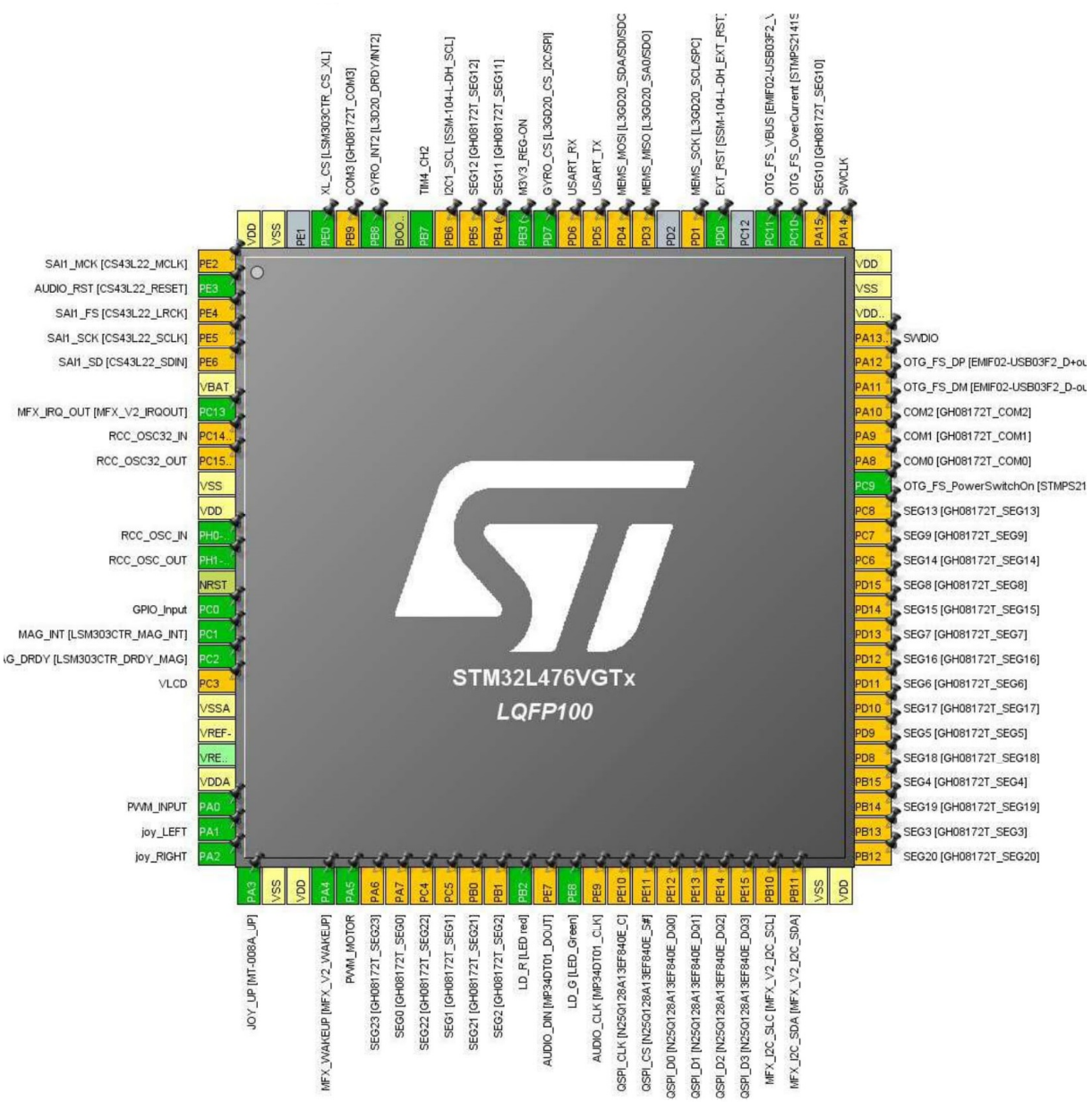
1 Opis projektu

Celem projektu jest zbudowanie zdalnie sterowanego robota jezdnego. Robot będzie sterowany za pomocą akcelerometru w telefonie. Dane będą przesyłanie za pomocą Wi-Fi lub Bluetooth. Regulacja prędkości będzie się odbywać za pomocą regulatora PID. Dane o prędkości będą pobierane z enkoderów znajdujących się w kołach robota. Opcjonalnie robot będzie wyświetlał szczegółowe dane o swoim stanie wewnętrznym za pomocą wbudowanego w płytkę z mikrokontrolerem wyświetlacza LCD.

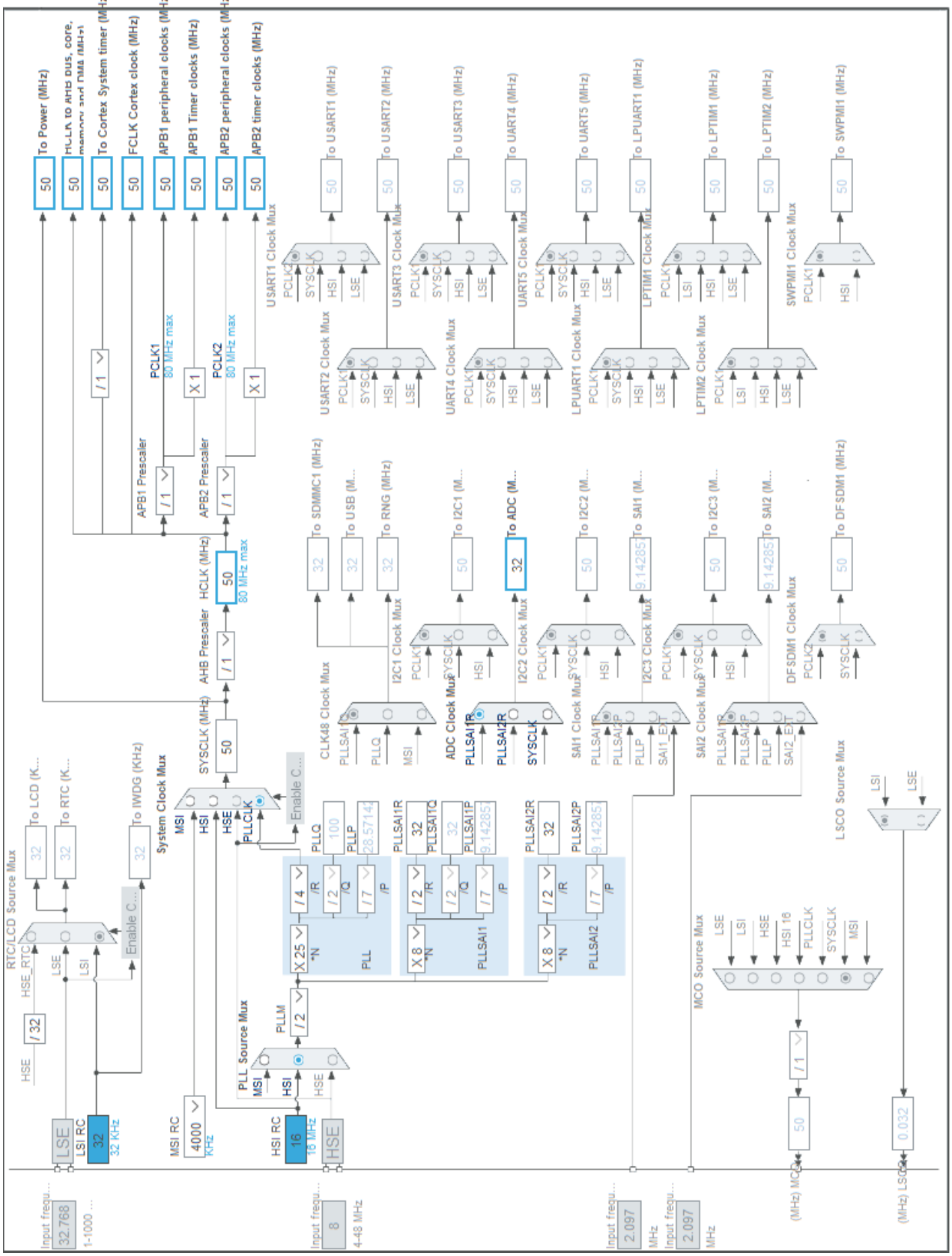


Rysunek 1: Architektura systemu

2 Konfiguracja mikrokontrolera



Rysunek 2: Konfiguracja wyjść mikrokontrolera w programie STM32CubeMX



Rysunek 3: Konfiguracja zegarów mikrokontrolera

2.1 Konfiguracja pinów

PIN	Tryb pracy	Funkcja/etykieta
PC14	OSC32_IN* RCC_OSC32_IN	
PC15	OSC32_OUT* RCC_OSC32_OUT	
PH0	OSC_IN* RCC_OSC_IN	
PH1	OSC_OUT*	RCC_OSC_OUT
PA2	USART2_TX	USART_TX
PA3	USART2_RX	USART_RX
PA0	TIM5 CH1	ENCODER_P
PA5	TIM2_CH1	DO_PRZODU
PB6	TIM4_CH1	SERVO

Tabela 1: Konfiguracja pinów mikrokontrolera

2.2 USART

Interfejs jest wykorzystywany do komunikacji z modułem Wi-Fi (ESP8266). Moduł odbiera dane za pomocą interfejsu UDP i przekazuje je do mikrokontrolera STM32 za pomocą interfejsu komunikacji szeregowej.

Parametr	Wartość
Baud Rate	115200
Word Length	8 Bits (including parity)
Parity	None
Stop Bits	1

Tabela 2: Konfiguracja peryferium USART

2.3 Timer 2

Parametr	Wartość
Clock Source	Internal Clock
Channel1	PWM Generation CH1
Prescaler	TIM2_PRESC
Counter Mode	Up
Counter Period	TIM2_PERIOD
Internal Clock Division	No Division
Mode	PWM mode 1
CH Polarity	High

Tabela 3: Konfiguracja peryferium Timer 2

2.4 Timer 4

Parametr	Wartość
Clock Source	Internal Clock
Channel	PWM Generation CH1
Prescaler	TIM4_PRESC
Counter Mode	Up
Counter Period	TIM4_PERIOD
Internal Clock Division	No Division
Mode	PWM mode 1
CH Polarity	High

Tabela 4: Konfiguracja peryferium Timer 4

2.5 Timer 5

Parametr	Wartość
Clock Source	Internal Clock
Channel	Input Capture direct mode
Prescaler	TIM5_PRESC
Counter Mode	Up
Counter Period	TIM5_PERIOD
Internal Clock Division	No Division

Tabela 5: Konfiguracja peryferium Timer 6

3 Urządzenia zewnętrzne

3.1 Moduł Wi-Fi ESP8266

Moduł dobiera dane za pomocą protokołu UDP i przekazuje je odpowiednio sformatowane za pomocą portu szeregowego do mikrokontrolera STM32. Do oprogramowania modułu wykorzystano framework Arduino.
Ustawienia komunikacji szeregowej:

Parametr	Wartość
Baud Rate	115200
Word Length	8 Bits (including parity)
Parity	None
Stop Bits	1

Tabela 6: Konfiguracja peryferium UART w module Wi-Fi

3.2 Smartfon z aplikacją RoboRemo

Przesyła wartości osi X i Y żyroskopu za pomocą protokołu UDP.
Ustawienia żyroskopu w aplikacji:

Parametr	Wartość
Etykieta	x
Wzmocnienie	1.0
Typ danych	int
Zakres	od -255 do 255
Limit do [min:max]	tak

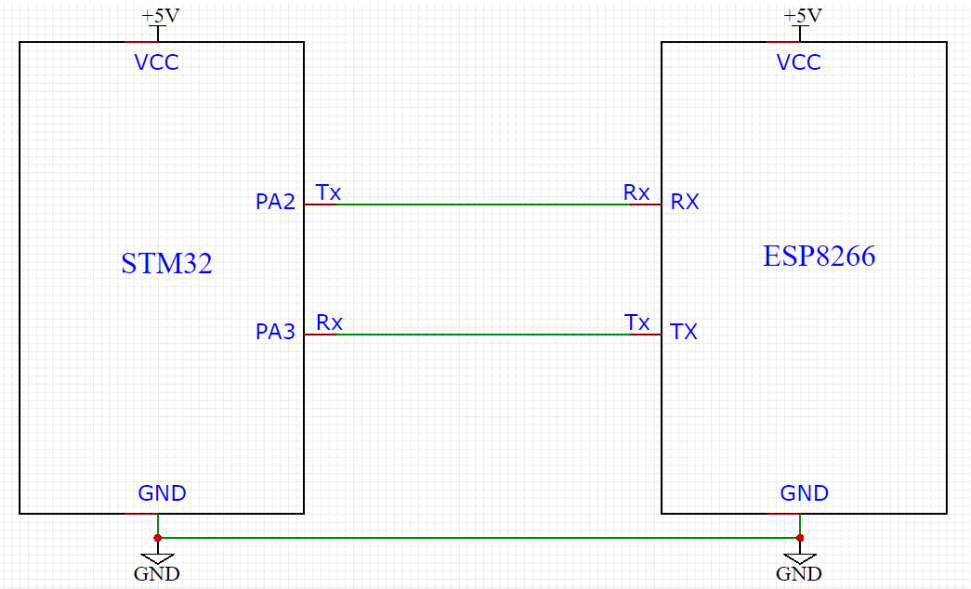
Tabela 7: Konfiguracja osi X

Parametr	Wartość
Etykieta	y
Wzmocnienie	0.5
Typ danych	int
Zakres	od 50 do 150
Limit do [min:max]	tak

Tabela 8: Konfiguracja osi Y

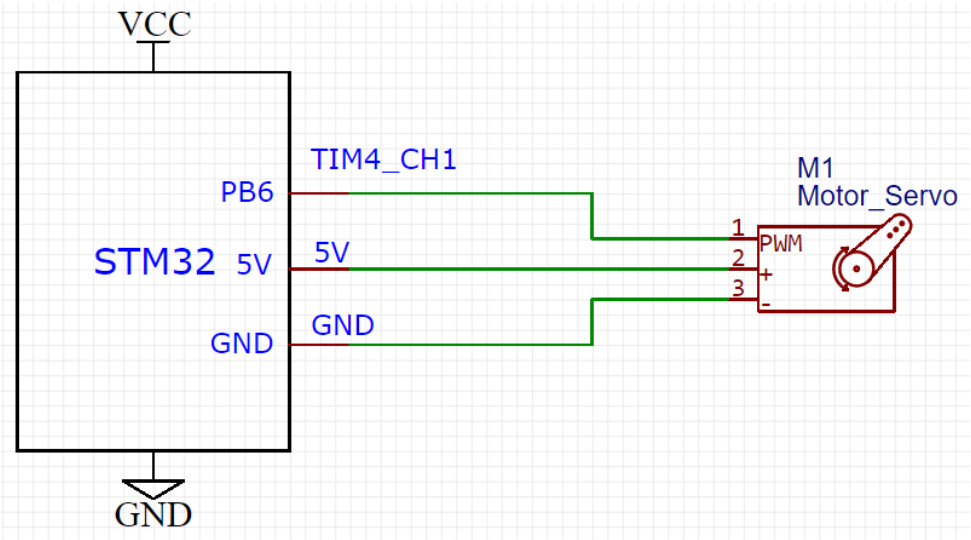
4 Projekt elektroniki

4.1 Połączenie pomiędzy STM32 a modułem WiFi



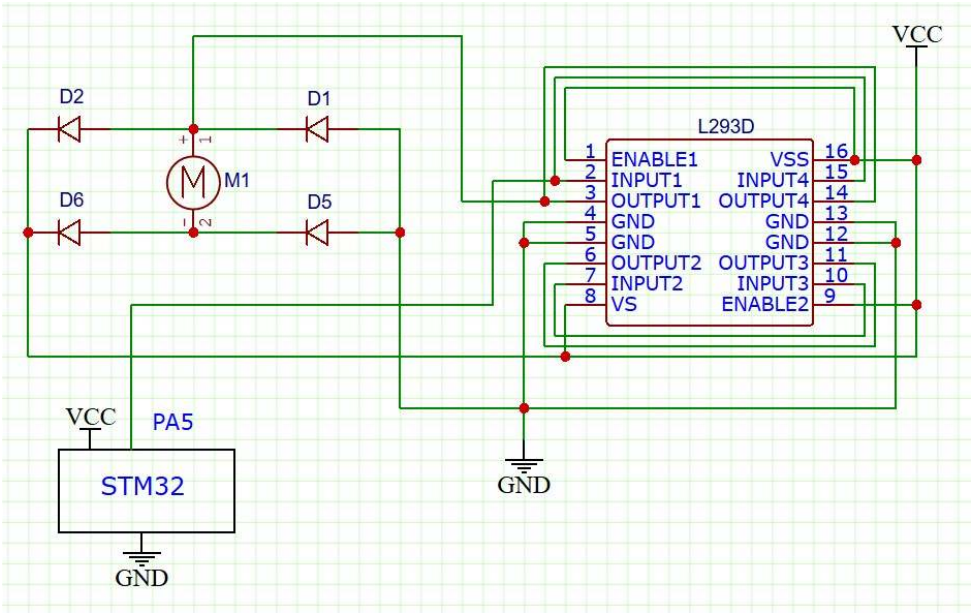
Rysunek 4: Połączenie STM32 z ESP8266

4.2 Schemat połączenia z serwomechanizmem



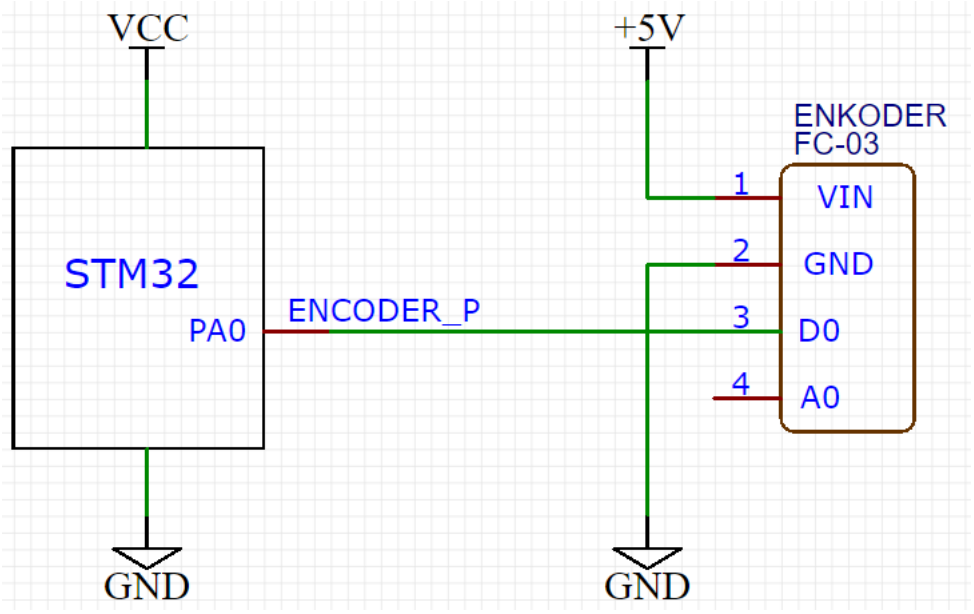
Rysunek 5: Połączenie STM32 z serwomechanizmem

4.3 Schemat połączenia z silnikiem



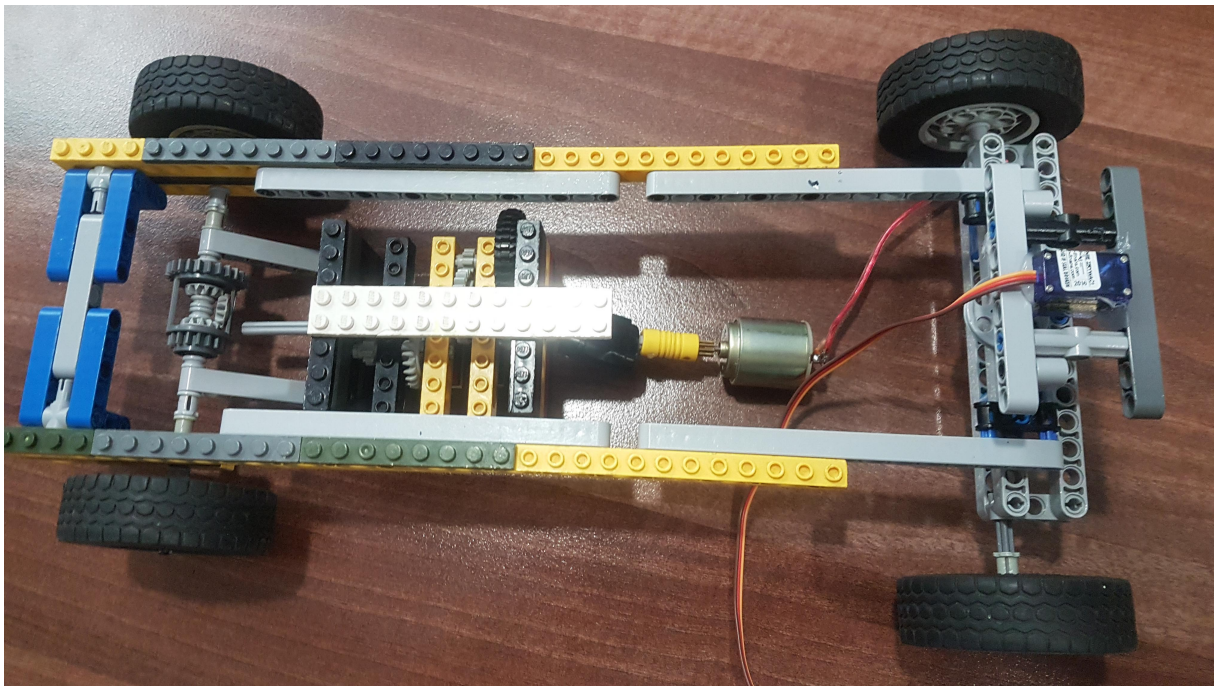
Rysunek 6: Schemat regulacji prędkości obrotowej silnika

4.4 Schemat połączenia z enkoderem

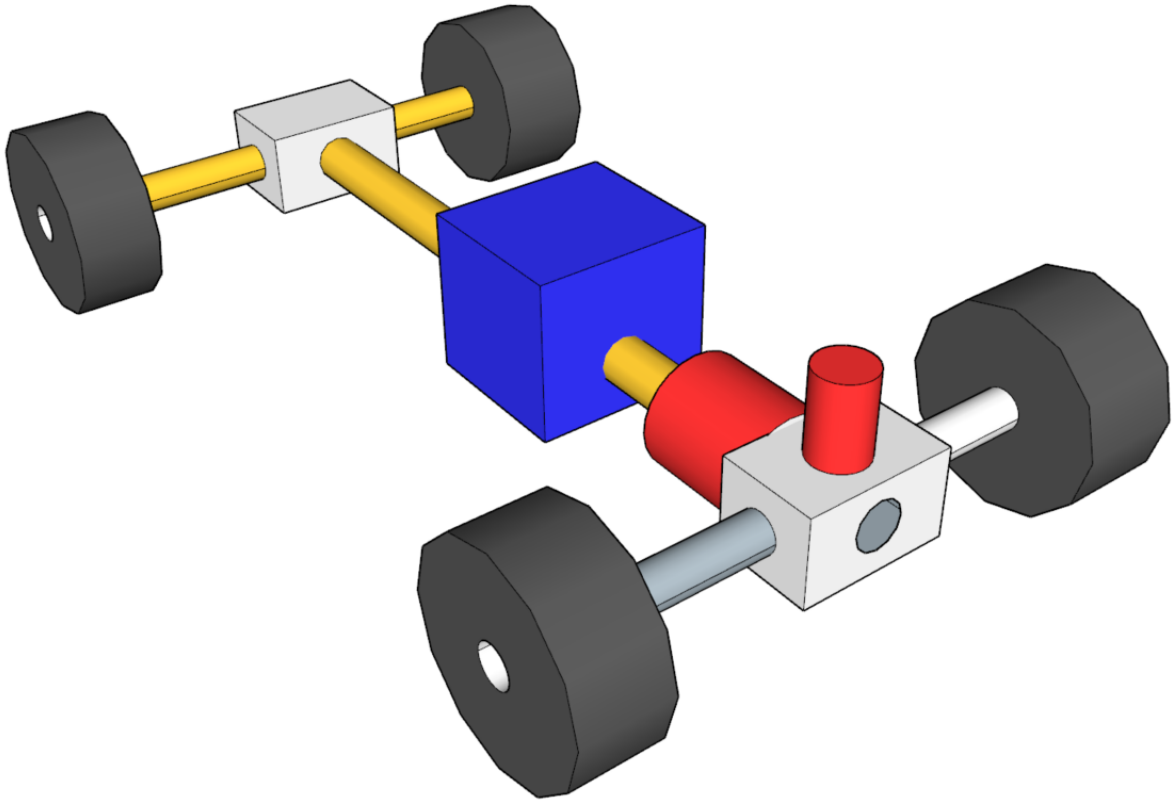


Rysunek 7: Schemat połączenia mikrokontrolera z enkoderem

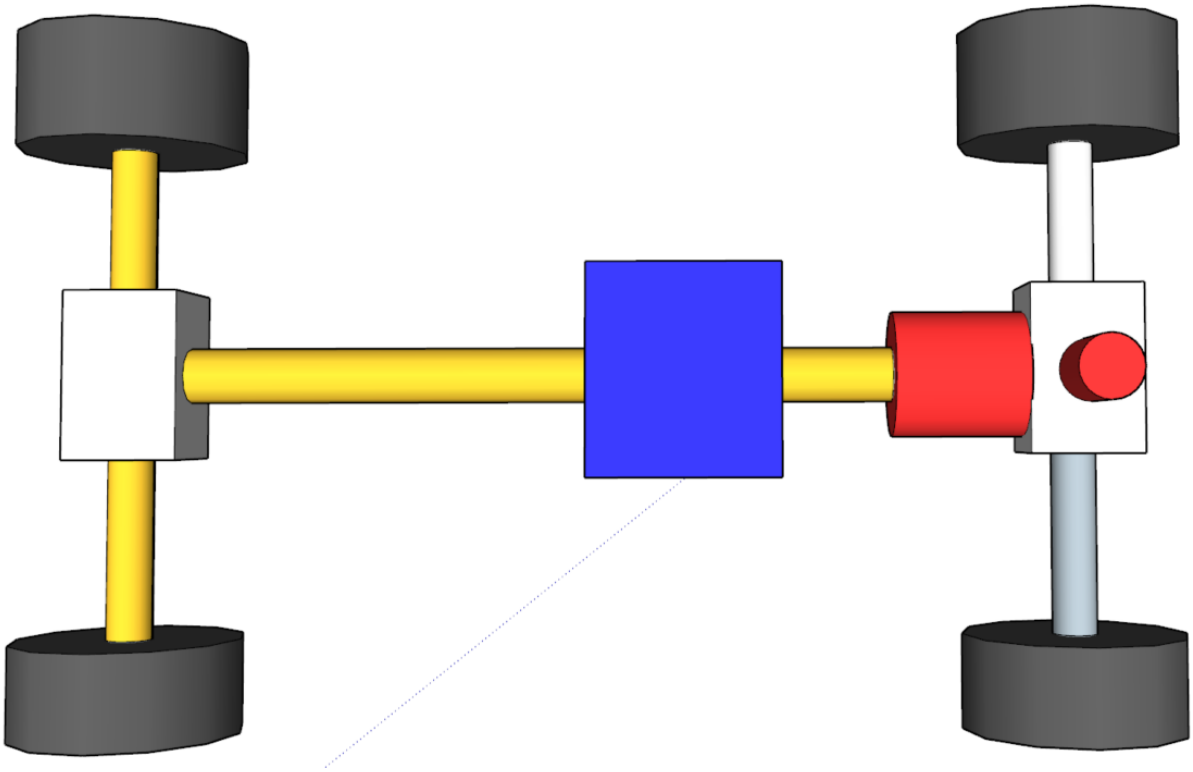
5 Konstrukcja mechaniczna



Rysunek 8: Zdjęcie części mechanicznej



Rysunek 9: Schemat mechaniczny 1/2



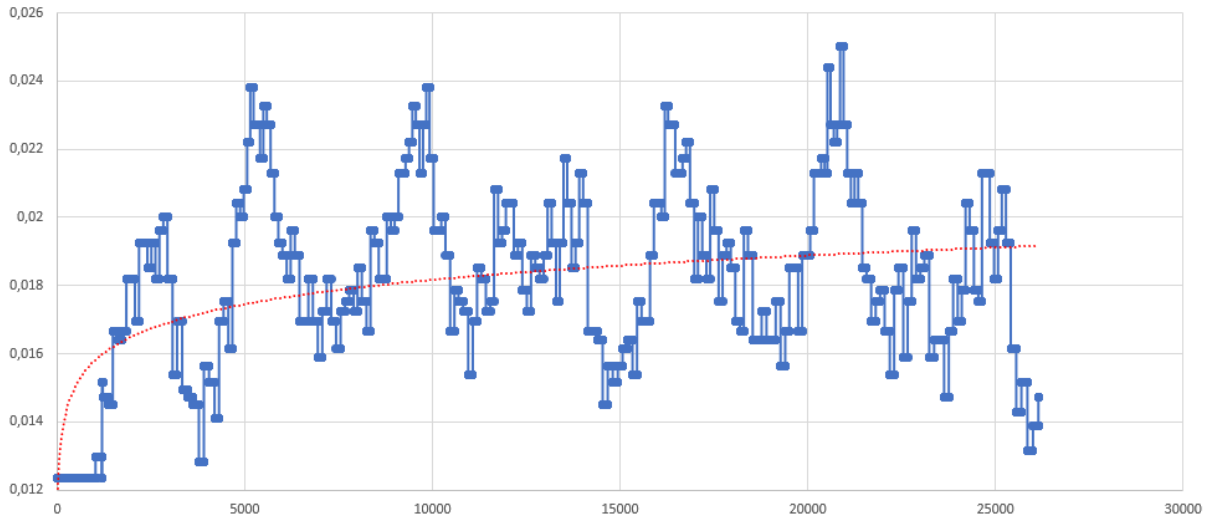
Rysunek 10: Schemat mechaniczny 2/2

Kolor	**Opis**
Czerwony	Silniki
Niebieski	Przekładnia
Żółty	Elementy przeniesienia napędu

Tabela 9: Legenda

6 Regulator PID

6.1 Wykres odpowiedzi skokowej



Rysunek 11: Odpowiedz skokowa

Na wykresie występuja bardzo duże oscylacje, które są spowodowane niedoskonałością wykonanego mechanizmu różnicowego oraz zbyt małą rozdzielczością odczytów z enkoderów. W związku z tym odpowiedź skokową przybliżono linią trendu i na jej podstawie za pomocą metody Zieglera Nicholsa wyliczono parametry K_p, K_i *oraz* K_d.

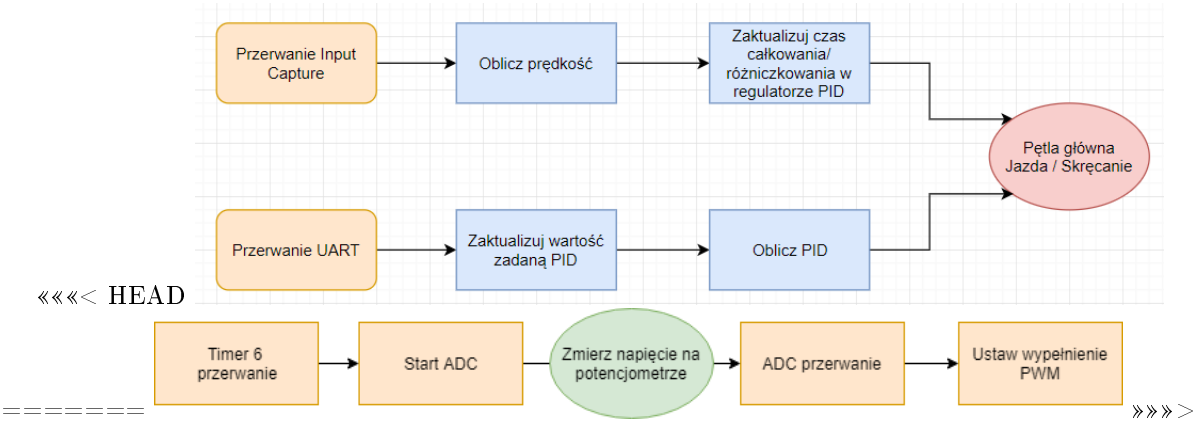
6.2 Wyznaczenie parametrów PID

$$\mathrm{k} = \Delta wy/\Delta we = \frac{0.01923}{0.01515} = 1.2693 \approx 1.27$$

$$\mathrm{K}_p = 0.6 * k = 0.762$$
$$K_i = 2 * \frac{K_p}{P_u} = 1.879$$
$$K_d = \frac{K_p * P_u}{8} = 0.086$$

7 Opis działania programu

7.1 Schemat działania programu



Rysunek 12: Schemat działania programu

7.2 Ręczna inicjalizacja parametrów w funkcji main

```
flag1 = flag2 = 0;
pwm_dutyL = pwm_dutyP = 0;
HAL_TIM_IC_Start_IT(&htim5, TIM_CHANNEL_1);
HAL_TIM_IC_Start_IT(&htim5, TIM_CHANNEL_2);

HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_1);
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_2);

HAL_UART_Receive_DMA(&huart2, &Received, BUFSIZE);
PIDInit(&pid, kp, ki, kd, sampleTimeSeconds, minOutput, maxOutput, mode, control
    );
PIDSetpointSet(&pid, set_value);
PIDInputSet(&pid, speed);
```

7.3 Funkcja obsługująca Input Capture

```
int speed;
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
  if (htim == &htim5)
  {
    if (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1)
    {
      period1 = __HAL_TIM_GET_COMPARE(&htim5, TIM_CHANNEL_1);
      __HAL_TIM_SET_COUNTER(&htim5, 0);
      flag1 = 1;
      speed = (period1 <= 50) ? (255 - map(period1, 20, 50, 0, 255)) : 0;
    }
    sampleTimeSeconds = period1 / 1000.0;
    PIDInit(&pid, kp, ki, kd, sampleTimeSeconds, minOutput, maxOutput, mode,
        control);
  }
}
```

7.4 Funkcja obsługująca przerwanie USART

```
volatile uint8_t xvalue, yvalue, yprevious, xprevious;
volatile char xsign;
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
  axis = (char)(Received[nrAxis]);
  sign = (char)(Received[nrSign]);
  value = (uint8_t)(Received[nrValue]);

  if(axis == 'x')
  {
    xsign = sign;
    xprevious = xvalue;
    xvalue = value;
  }

  if(axis == 'y')
  {
    yprevious = yvalue;
    yvalue = value;
    if(abs(value - xprevious) < 2)
      yvalue = yprevious;
    else if(value > 100)
      yvalue = yprevious;
    else if(value < 50)
      yvalue = yprevious;
  }
  PIDSetpointSet(&pid, set_value);
  PIDInputSet(&pid, speed);
  PIDCompute(&pid);
  pid_output = PIDOutputGet(&pid);
  HAL_UART_Receive_DMA(&huart2, &Received, BUFSIZE);
}
```

7.5 Funkcja realizująca jazdę

```
void Jazda()
{
  __HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, 255 - pid_output);
}
```

7.6 Funkcja realizująca skręcanie

```
void Skrecanie()
{
  __HAL_TIM_SET_COMPARE(&htim4, TIM_CHANNEL_1, yvalue);
}
```

7.7 Funkcja mapująca wartości czasowe na wartość zadaną regulatora PID

```
long map(long x, long in_min, long in_max, long out_min, long out_max)
{
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```

7.8 Funkcja przekierowująca printf()

```
int _write(int file, char *s, int len)
{
  HAL_UART_Transmit_DMA(&huart2, (uint8_t*)s, len);
  return len;
}
```

7.9 Pętla główna

```
while (1)
{
  Jazda();
  Skrecanie();
}
```

8 Zadania niezrealizowane

9 Podsumowanie

«««< HEAD

Literatura

[1] Krzysztof Amborski, Andrzej Murusak: *Teoria sterowania w ćwiczeniach*, ('78)

[2] Józef Lisowski: *Podstawy automatyki*, (2015)

[3] Jerzy Brzózka: *Regulatory i układy automatyki*, (2004)

[4] Krzysztof Tchoń: *Manipulatory i roboty mobilne : modele, planowanie ruchu, sterowanie*, (2000)

[5] Stanisław Wszelak: *Administrowanie sieciowymi protokołami komunikacyjnymi*, (2015)

=======

Literatura

[1] www.forbot.pl, *Kurs STM32 F1 HAL – 7 – liczniki (timery) w praktyce, PWM*

[2] www.forbot.pl, *Jak zaimplementować regulator PID dla silnika z enkoderem?* »»»> e36a3efe20562338b358633776b369084afd4f0f